INFORMATION AND COMPUTATION **139**, 39–56 (1997)
ARTICLE NO. IC972662

Termination of System *F*-bounded: A Complete Proof

Giorgio Ghelli

Dipartimento di Informatica, Università di Pisa, Corso Italia 40, I-56125 Pisa, Italy
E-mail: ghelli@di.unipi.it

System *F*-bounded is a second-order typed λ-calculus with subtyping which has been defined to carry out foundational studies about the type systems of object-oriented languages. The almost recursive nature of the essential feature of this system makes one wonder whether it retains the strong normalization property, with respect to first- and second-order $\beta\eta$ reduction of system $F_{\leqslant}$. We prove that this is the case. The proof is carried out to the last detail to allow the reader to be convinced of its correctness.

1. INTRODUCTION

System *F*-bounded is a second-order lambda calculus with subtyping. It extends system $F_{\leqslant}$ (see [Cardelli Wegner 85, Curien Ghelli 92, Cardelli *et al.* 94, Ghelli 90]) in that the bound of a type variable may contain the variable itself. In object-oriented terms, this extension allows one to write functions which accept parameters belonging to all the classes which inherit from one class. *F*-bounded quantification was introduced in [Canning *et al.* 89] and is currently included in, or at the basis of, many proposals for strongly typed object-oriented languages (e.g., [Bruce 94, Mitchell 90a, Katiyar *et al.* 94, Eifrig *et al.* 94, Bruce *et al.* 95]); see [Abadi Cardelli 95, Bruce *et al.* 96, Fisher Mitchell 94, Pierce Turner 94] for general and perceptive discussions about the design of type systems for object-oriented languages.

A typed λ-calculus is *strongly normalizing* (or *terminating*) when no infinite reduction chain starts from a typed term of that calculus. Termination is related to the possibility to solve some recursive type equations. For example, in a system with subtyping, if the disequation system $\{\alpha \leqslant \alpha \to \alpha, \alpha \to \alpha \leqslant \alpha\}$ has a solution, then the nonterminating term $(\lambda x{:}\,\alpha.x(x))(\lambda x{:}\,\alpha.x(x))$ is well typed. The second disequation can already be solved in $F_{\leqslant}$, for example by $\alpha = \text{Top}$. *F*-bounded quantification makes it possible to solve the first disequation too ($\alpha = t$ solves $\alpha \leqslant \alpha \to \alpha$, if t is *F*-bounded by $t \to t$), and others which have no solution in $F_{\leqslant}$. This raises the question of whether the addition of *F*-bounded quantification may allow

0890-5401/97 $25.00

nonterminating terms to be written. In this paper we prove that such terms cannot, in fact, be written.

Our proof is based on Tait and Girard's method of saturated sets [Tait 76, Girard 72], and our approach is similar to the one used in [Mitchell 86] to prove termination for F, and to the proof of termination of $\beta\eta$ reduction for $F_{\leqslant}$ given in [Ghelli 90]. We first show that the type erasure of each well-typed F-bounded term is strongly normalizing. To this end, we define an interpretation of F-bounded types such that each type is interpreted with a superset of the type erasures of all F-bounded terms with that type, and this superset is small enough to be contained in the set SN of strongly normalizing λ-terms: $\Gamma \vdash a : T \Rightarrow (\text{typeErasure}(a) \in [\![\Gamma \vdash T]\!]$ and $[\![\Gamma \vdash T]\!] \subseteq \text{SN})$. From this, we derive strong normalization for system F-bounded. The proof is carried out in full detail, since many subtle errors have been discovered in termination proofs, and in proofs dealing with systems related to $F_{\leqslant}$. Hence, we felt that this strong normalization property, which we find quite surprising, deserved the writing of a full proof, which can be checked by the reader. Another proof is contained in [McAllester *et al.* 95], which was announced soon after the writing of this paper; however, the two papers are very different. [McAllester *et al.* 95] describes a general technique to prove termination for a wide range of systems and hints how it may be applied to a type-inference (Curry-style) version of F-bounded as well. Our proof, on the other hand, only proves termination for explicitly types (Church-style) F-bounded and for its subsystems (say, $F_{\leqslant}$), but it is complete and developed to the last detail, which would not be reasonable if it were part of a paper written with a broader scope. In this paper the reader can also find the only full formalization of system F-bounded that we are aware of.

In [Geuvers 93, Goguen 94, Mitchell 90b], as in [McAllester *et al.* 95], the Tait–Girard method is generalized to prove other properties of reduction, namely confluence, and to factorize the similarities in the termination proofs of different typed calculi. These papers do not deal with subtyping, but it would not be difficult to extend their techniques for this purpose. We have chosen a direct assault on F-bounded normalization, rather than going through such general techniques, since this is the easiest approach when one is interested in just one property for one system ($\beta\eta$ reduction for system F-bounded does not enjoy confluence, for the reasons outlined in [Curien Ghelli 94]).

System F-bounded is introduced in Section 2. Strong normalization is proved in Section 3.

2. SYSTEM *F*-BOUNDED

We adopt the following syntax for F-bounded types, terms, environments, and judgements.

PreTypes	$A ::= t \mid \text{Top} \mid A \to A \mid \forall t \leqslant A.A$
PreTermes	$a ::= x \mid \lambda x\colon A.a \mid a(a) \mid \Lambda t \leqslant A.a \mid a\{A\}$
Pre-Type Environments	$\Gamma ::= () \mid \Gamma, t \leqslant A$

Pre-Value Environments $\Delta ::= () \mid \Delta, x: A$

Pre-Name Environments $E ::= () \mid E, t$

Pre-Judgments $J ::= E \vdash \diamond \mid \Gamma \vdash \diamond \mid \Gamma, \Delta \vdash \diamond \mid E \vdash A$

$\mid \Gamma \vdash A \leqslant A \mid \Gamma, \Delta \vdash a: A.$

Types, terms, and judgements are as in system $F_{\leqslant}$ (see [Curien Ghelli 92, Cardelli *et al.* 94]), with some small differences which we will now discuss. $\forall t \leqslant A.B$ is the type of a function which expects a type A' which is a subtype of $A[t \leftarrow A']$ and returns a value whose type is $B[t \leftarrow A']$, while in system $F_{\leqslant}$ t cannot appear in A. Type and value environments (Γ and Δ) assign upper bounds and types to type variables and value variables, similarly to what happens in most $F_{\leqslant}$ presentations, with the small difference that we have kept the two environments separate. A name environment E is a set of type variables which is useful in the $E \vdash A$ judgement, which can be read as "A is well formed in E" or as "every free variable of A is in E"; we will comment on this later. $E/\Gamma/\Gamma, \Delta \vdash \diamond$ are good formation judgements for name/type/value environments. These judgements state that no variable is defined twice in the environment and that every type variable which is free in a bound in Γ, or in a type in Δ, has been previously defined in Γ.

System *F*-bounded has been introduced to deal with the problem of "binary methods"; we will hint here at the basic idea and refer to [Canning *et al.* 89, Bruce *et al.* 96, Fisher Mitchell 94] for more details. A binary method of an object type T is a method with an argument whose type is T itself. Consider two object types $A = \mu t.[a: \text{Int}; eq: t \to \text{Bool}]$ and $B = \mu t.[a: \text{Int}; eq: t \to \text{Bool}; b: \text{Int}]$; here $\mu t.T$ defines a recursive type (i.e., t in T stands for $\mu t.T$ itself), and $[a: A; \ldots]$ is a record type. Because of the binary method eq, B is not a subtype of A (see [Amadio Cardelli 93]), hence we cannot write a function which operates on objects of both types in system $F_{\leqslant}$. However, both A and B satisfy the condition $t \leqslant [a: \text{Int}; eq: t \to \text{Bool}]$, hence in system *F*-bounded a function with type $\forall t \leqslant [a: \text{Int}; eq: t \to \text{Bool}]$. $t \to \cdots$ can be safely applied to objects of both types. *F*-bounded quantification is also the basis to allow B to be defined by inheritance from A. In essence, if we type-check the A methods in an environment where "$t \leqslant [a: \text{Int}: eq: t \to \text{Bool}]$, self: t", then we can safely inherit those methods in B, while in system $F_{\leqslant}$ we cannot define methods which work for both A and B, since we have no good type for *self*.

As usual, we only prove termination for a recursion-free version of system *F*-bounded, though to make any use of system *F*-bounded, some form of recursion is clearly needed. There is no contradiction in this. In a sense, we study the termination of the recursion-free part of the language because we want to understand whether adding recursion to system *F*-bounded would make it essentially different or not.[1]

We now give the formal presentation of the system.

[1] More precisely, if we consider types as propositions, termination proves that the recursion-free kernel of the type system is "logically consistent" (see [Goguen 94]), while it is known that the kind of recursive values and types that we need in object-oriented programming are not logically consistent.

Free type variables.

$$\textbf{Types:}\quad \mathrm{FTV}(t)=\{t\};\qquad \mathrm{FTV}(\mathrm{Top})=\{\};$$

$$\mathrm{FTV}(A\to A')=\mathrm{FTV}(A)\cup\mathrm{FTV}(A');$$

$$\mathrm{FTV}(\forall t\leqslant A.A')=(\mathrm{FTV}(A)\cup\mathrm{FTV}(A'))\backslash\{t\}$$

$$\textbf{Value Environments:}\quad \mathrm{FTV}(())=\{\};\ \mathrm{FTV}(x\colon A,\Delta)=\mathrm{FTV}(A)\cup\mathrm{FTV}(\Delta)$$

$$\textbf{Type Environments:}\quad \mathrm{FTV}(())=\{\};\ \mathrm{FTV}(t\leqslant A,\Gamma)=(\mathrm{FTV}(A)\cup\mathrm{FTV}(\Gamma))\backslash\{t\}.$$

Notation 2.1 ($t\in\Gamma$). We write $t\in\Gamma$ if $t\leqslant A$ is in Γ, for some A; similarly for $x\in\Delta$ and $t\in E$.

Notation 2.2. ($\mathrm{vars}(\Gamma)$). $\mathrm{vars}(t_1\leqslant A_1,\ldots,t_n\leqslant A_m)=t_1,\ldots,t_n$.

Name environment, type environment, and value environment formation rules.

$$(\varnothing\ \text{NameEnv})\quad ()\vdash\diamond \qquad (\text{NameEnv})\quad \frac{E\vdash\diamond\quad t\notin E}{E,t\vdash\diamond}$$

$$(\varnothing\ \text{TypeEnv})\quad ()\vdash\diamond \qquad (\text{TypeEnv})\quad \frac{\Gamma\vdash\diamond\quad \mathrm{vars}(\Gamma),t\vdash A\quad t\notin\Gamma}{\Gamma,t\leqslant A\vdash\diamond}$$

$$(\varnothing\ \text{ValueEnv})\quad \frac{\Gamma\vdash\diamond}{\Gamma,()\vdash\diamond} \qquad (\text{ValueEnv})\quad \frac{\Gamma,\Delta\vdash\diamond\quad \mathrm{vars}(\Gamma)\vdash A\quad x\notin\Delta}{\Gamma,\Delta,x\colon A\vdash\diamond}$$

Type formation rules.

$$(\text{Var Form})\quad \frac{E,t,E'\vdash\diamond}{E,t,E'\vdash t} \qquad (\text{Top Form})\quad \frac{E\vdash\diamond}{E\vdash\mathrm{Top}}$$

$$(\to\ \text{Form})\quad \frac{E\vdash A\quad E\vdash B}{E\vdash A\to B} \qquad (\forall\ \text{form})\quad \frac{E,t\vdash A\quad E,t\vdash B}{E\vdash\forall t\leqslant A.B}$$

Subtype rules.

$$(\text{Id}\leqslant)\quad \frac{\Gamma\vdash\diamond\quad \mathrm{vars}(\Gamma)\vdash A}{\Gamma\vdash A\leqslant A} \qquad (\text{Trans}\leqslant)\quad \frac{\Gamma\vdash A\leqslant B\quad \Gamma\vdash B\leqslant C}{\Gamma\vdash A\leqslant C}$$

$$(\text{Var}\leqslant)\quad \frac{\Gamma,t\leqslant A,\Gamma'\vdash\diamond}{\Gamma,t\leqslant A\,\Gamma'\vdash t\leqslant A} \qquad (\text{Top}\leqslant)\quad \frac{\Gamma\vdash\diamond\quad \mathrm{vars}(\Gamma)\vdash A}{\Gamma\vdash A\leqslant\mathrm{Top}}$$

$$(\to\ \leqslant)\quad \frac{\Gamma\vdash A\leqslant A'\quad \Gamma\vdash B\leqslant B'}{\Gamma\vdash A'\to B<A\to B'} \qquad (\forall\leqslant)\quad \frac{\Gamma,t\leqslant A\vdash t\leqslant A'\quad \Gamma,t\leqslant A\vdash B\leqslant B'}{\Gamma\vdash\forall t\leqslant A'.B\leqslant\forall t\leqslant A.B'}$$

Typing rules.

$$\text{(Var)}\quad \frac{\Gamma, \Delta, x\colon A, \Delta' \vdash \diamond}{\Gamma, \Delta, x\colon A, \Delta' \vdash x\colon A} \qquad \text{(Subsump)}\quad \frac{\Gamma, \Delta \vdash a\colon A \quad \Gamma \vdash A \leqslant B}{\Gamma, \Delta \vdash a\colon B}$$

$$(\rightarrow \text{Intro})\quad \frac{\Gamma, \Delta, x\colon A \vdash b\colon B}{\Gamma, \Delta \vdash \lambda x\colon A.b\colon A \rightarrow B} \qquad (\rightarrow \text{Elim})\quad \frac{\Gamma, \Delta \vdash f\colon A \rightarrow B \quad \Gamma, \Delta \vdash a\colon A}{\Gamma, \Delta \vdash f(a)\colon B}$$

$$(\forall \text{ Intro})\quad \frac{\Gamma, t \leqslant A, \Delta \vdash b\colon B \quad t \notin \mathrm{FTV}(\Delta)}{\Gamma, \Delta \vdash \Lambda t \leqslant A.b\colon \forall t \leqslant A.B} \qquad (\forall \text{ Elim})\quad \frac{\Gamma, \Delta \vdash f\colon \forall t \leqslant A.B \quad \Gamma \vdash A' \leqslant A[t \leftarrow A']}{\Gamma, \Delta \vdash f\{A'\}\colon B[t \leftarrow A']}$$

Reduction rules.

$$\begin{array}{lll} (\beta) & (\lambda x\colon A.b)(a) & \longrightarrow b[x \leftarrow a] \\ (\eta) & \lambda x\colon A.b(x) & \longrightarrow b \qquad x \notin \mathrm{FV}(b) \\ (\beta 2) & (\Lambda t \leqslant A.b)\{A'\} & \longrightarrow b[t \leftarrow A'] \\ (\eta 2) & \Lambda t \leqslant A.b\{t\} & \longrightarrow b \qquad t \notin \mathrm{FTV}(b) \end{array}$$

Observe that, in $F_{\leqslant}$, an environment $\Gamma, t \leqslant A$ is well formed if Γ is well formed, t is not defined in $\Gamma (t \notin \Gamma)$, and every free variable in A is defined in Γ (Fig. 1), while in *F*-bounded we relax the last condition and require that every free variable in A be defined in $\Gamma, t \leqslant A$. In the traditional $F_{\leqslant}$ style presentation, this fact may be formalized by the rule ($F_{\leqslant}$-like TypeEnv) in Fig. 1, but then, to prove the premise $\Gamma, t \leqslant A \vdash A$, we would need to prove $\Gamma, t \leqslant A \vdash \diamond$ itself. We avoid this problem by formalizing the condition that the free variables of A are defined in $(\Gamma, t \leqslant A)$ as "vars(Γ), $t \vdash A$" (rule (TypeEnv) above); this judgement does not depend on the irrelevant information contained in the bounds in Γ, and its proof does not depend on $\Gamma, t \leqslant A \vdash \diamond$. A different solution would be to reduce $\Gamma, t \leqslant A \vdash \diamond$ to $(\Gamma \vdash \diamond, t \notin \Gamma, \Gamma, t \leqslant \mathrm{Top} \vdash A)$, if $A \neq \mathrm{Top}$, and to $(\Gamma \vdash \diamond, t \notin \Gamma)$ only, if $A = \mathrm{Top}$. This is slightly less elegant, but avoids name environments; here we have adopted the former approach since it makes our proof a little bit shorter.

Rule $(\forall \leqslant)$ is slightly more powerful than the rule ($F_{\leqslant}$-like $\forall \leqslant$) (Fig. 2), which would be the immediate generalization of the $F_{\leqslant}$ rule. The $F_{\leqslant}$-like rule is admissible in our system (it is less powerful than $(\forall \leqslant)$): whenever we can prove the premise $\Gamma, t \leqslant A \vdash A \leqslant A'$, the premise $\Gamma, t \leqslant A \vdash t \leqslant A'$ of rule $(\forall \leqslant)$ can be proved too, by applying the $(\mathrm{Var} \leqslant)$ rule, to obtain $t \leqslant A$, and then transitivity. On the other hand, the rule we have chosen allows us to prove that both $\forall t \leqslant t.A \leqslant \forall t \leqslant \mathrm{Top}.A$ and $\forall t \leqslant \mathrm{Top}.A \leqslant \forall t \leqslant t.A$ hold, while in the $F_{\leqslant}$-like system only $\forall t \leqslant \mathrm{Top}.A \leqslant \forall t \leqslant t.A$ holds. This double inclusion is a desirable property, since

$$(\mathrm{F}_{\leqslant}\text{TEnv})\quad \frac{\Gamma \vdash \Diamond \quad \Gamma \vdash \mathrm{A} \quad \mathrm{t} \notin \Gamma}{\Gamma, \mathrm{t} \leqslant \mathrm{A} \vdash \Diamond} \qquad (\mathrm{F}_{\leqslant}\text{-like TEnv})\quad \frac{\Gamma \vdash \Diamond \quad \Gamma, \mathrm{t} \leqslant \mathrm{A} \vdash \mathrm{A} \quad \mathrm{t} \notin \Gamma}{\Gamma, \mathrm{t} \leqslant \mathrm{A} \vdash \Diamond}$$

FIG. 1. An $F_{\leqslant}$-like version of rule (TypeEnv).

$$(F_{\leqslant}\ \forall\leqslant)\ \frac{\Gamma \vdash A \leq A' \quad \Gamma, t{\leqslant}A \vdash B \leq B'}{\Gamma \vdash \forall t{\leqslant}A'.B \leq \forall t{\leqslant}A.B'} \qquad (F_{\leqslant}\text{-like}\ \forall\leqslant)\ \frac{\Gamma, t{\leqslant}A \vdash A \leq A' \quad \Gamma, t{\leqslant}A \vdash B \leq B'}{\Gamma \vdash \forall t{\leqslant}A'.B \leq \forall t{\leqslant}A.B'}$$

FIG. 2. An $F_{\leqslant}$-like version of rule $(\forall\leqslant)$.

both bounds, $t \leqslant \mathrm{Top}$ and $t \leqslant t$, express the fact that t can be substituted by any type. We claim that this is the only difference between the two versions of the system, which would collapse if we either added a $\forall t \leqslant t.A \leqslant \forall t \leqslant \mathrm{Top}.A$ axiom to the $F_{\leqslant}$-like version, or if we forbade $\forall t \leqslant t.A$ types in our version; these observations and claims were first written in [Katiyar 92]. By adopting the strongest rule and allowing $\forall t \leqslant t.A$ types, we ensure that our strong normalization result still holds if a smaller language or a stricter rule are chosen.

3. STRONG NORMALIZATION OF *F*-BOUNDED TERMS

3.1. *Saturated Sets*

Before giving the strong normalization proof, the (standard) notion of saturated set must be introduced.

Notation 3.1 (Λ, SN, $\mathscr{P}$). Λ is the set of all (untyped) λ-terms (defined, as usual, as $a ::= x \mid \lambda x.a \mid aa$).

SN is the set of all $\beta\eta$ strongly normalizing λ-terms.

$\mathscr{P}(A)$ is the set of all subsets of A.

DEFINITION 3.2 (Saturated set). A set $R \subseteq \Lambda$ is saturated when:

Sat_0 $R \subseteq \mathrm{SN}$

Sat_1 $a \in \mathrm{SN}, (b[x \leftarrow a]b_1 \ldots b_n) \in R \Rightarrow (\lambda x.)ab_1 \ldots b_n \in R \qquad (n \geqslant 0)$

Sat_2 $b_1, \ldots, b_n \in \mathrm{SN} \qquad \Rightarrow xb_1 \ldots b_n \in R \qquad (n \geqslant 0).$

Notation 3.3 (SAT). SAT is the set of all saturated sets. Note that $\mathrm{SAT} \subseteq \mathscr{P}(\mathrm{SN}) \subseteq \mathscr{P}(\Lambda)$.

Remark 3.4 (NotEmpty). By Sat_2, if $R \in \mathrm{SAT}$, then, for any variable x, $x \in R$, hence $R \neq \varnothing$.

LEMMA 3.5 (SN). $\mathrm{SN} \in \mathrm{SAT}$.

Proof. We prove that Sat_1 and Sat_2 hold for SN.

Sat_1: $a \in \mathrm{SN}, b[x \leftarrow a]\, b_1 \ldots b_n \in \mathrm{SN} \Rightarrow (\lambda x.b)\, ab_1 \ldots b_n \in \mathrm{SN}.$

Proof of Sat_1. Let $\mathrm{depth}(a)$ be the maximum length of a $\beta\eta$ reduction chain starting from a term $a \in \mathrm{SN}$. We prove Sat_1 by induction on $\mathrm{depth}(b[x \leftarrow a]\, b_1 \ldots b_n) + \mathrm{depth}(a)$. Let $b[x \leftarrow a]\, b_1 \ldots b_n \in \mathrm{SN}$ and consider any reduction chain starting from $(\lambda x.b)\, ab_1 \ldots b_n$. The first step in this chain is one of the following:

(1)	$(\lambda x.b)\, ab_1 \ldots b_n$	$\longrightarrow_\beta$	$b[x \leftarrow a]\, b_1 \ldots b_n$	
(2)	$(\lambda x.b)\, ab_1 \ldots b_n$	$\longrightarrow_\eta$	$cab_1 \ldots b_n$	with $b = cx$ and $x \notin \mathrm{FV}(c)$
(3)	$(\lambda x.b)\, ab_1 \ldots b_n$	$\longrightarrow$	$(\lambda x.b')\, ab_1 \ldots b_n$	with $b \longrightarrow\!\triangleright\, b'$
(4)	$(\lambda x.b)\, ab_1 \ldots b_i \ldots b_n$	$\longrightarrow$	$(\lambda x.b)\, ab_1 \ldots b_i' \ldots b_n$	$1 \leqslant i \leqslant n$, $b_i \longrightarrow\!\triangleright\, b_i'$
(5)	$(\lambda x.b)\, ab_1 \ldots b_n$	$\longrightarrow$	$(\lambda x.b)\, a'b_1 \ldots b_n$	with $a \longrightarrow\!\triangleright\, a'$.

We show that in any case the reduced term is in SN. If $\mathrm{depth}(b[x \leftarrow a]\, b_1 \ldots b_n) + \mathrm{depth}(a) = 0$, only the first two cases are possible, and in those cases the inductive hypothesis is not needed.

(1) $b[x \leftarrow a]\, b_1 \ldots b_n \in \mathrm{SN}$ by hypothesis.

(2) By $b = cx$ and $x \notin \mathrm{FV}(b)$, $cab_1 \ldots b_n = b[x \leftarrow a]\, b_1 \ldots b_n$.
$b[x \leftarrow a]\, b_1 \ldots b_n \in \mathrm{SN}$ by hypothesis.

(3) $b \longrightarrow\!\triangleright\, b'$ implies $b[x \leftarrow a]\, b_1 \ldots b_n \to b'[x \leftarrow a]\, b_1 \ldots b_n$.
Hence $b'[x \leftarrow a]\, b_1 \ldots b_n \in \mathrm{SN}$.

Since $\mathrm{depth}(b'[x \leftarrow a]\, b_1 \ldots b_n) + \mathrm{depth}(a) < \mathrm{depth}(b[x \leftarrow a]\, b_1 \ldots b_n) + \mathrm{depth}(a)$, then, by induction:

$$b'[x \leftarrow a]\, b_1 \ldots b_n \in \mathrm{SN} \Rightarrow (\lambda x.b')\, ab_1 \ldots b_n \in \mathrm{SN}.$$

(4) The same as (3), but substitute b' with b and b_i with b_i'.

(5) $a \longrightarrow\!\triangleright\, a'$ implies that $b[x \leftarrow a]\, b_1 \ldots b_n$ reduces to $b[x \leftarrow a']\, b_1 \ldots b_n$ in $0 - n$ steps. Hence, $b[x \leftarrow a']\, b_1 \ldots b_n \in \mathrm{SN}$.

Since $\mathrm{depth}(b[x \leftarrow a']\, b_1 \ldots b_n) + \mathrm{depth}(a') < \mathrm{depth}(b[x \leftarrow a]\, b_1 \ldots b_n) + \mathrm{depth}(a)$, then, by induction:

$$b[x \leftarrow a']\, b_1 \ldots b_n \in \mathrm{SN} \Rightarrow (\lambda x.b)\, a'b_1 \ldots b_n \in \mathrm{SN}.$$

The addendum $\mathrm{depth}(a)$ is important when x is not free in b.

Sat_2: $b_1, \ldots, b_n \in \mathrm{SN} \Rightarrow xb_1 \ldots b_n \in \mathrm{SN}$

Proof of Sat_2. By induction on $\sum_{i=1..n} \mathrm{depth}(b_i)$. If $\sum_{i=1..n} \mathrm{depth}(b_i) = 0$, then $xb_1 \ldots b_n \in \mathrm{SN}$. Otherwise, consider any reduction chain starting from $xb_1 \ldots b_n$: $xb_1 \ldots b_i \ldots b_n \longrightarrow\!\triangleright\, xb_1 \ldots b_i' \ldots b_n \longrightarrow\!\triangleright\, \ldots$: b_i' is a reduct of b_i, hence $b_i' \in \mathrm{SN}$, and $\mathrm{depth}(b_i') < \mathrm{depth}(b_i)$. We can now apply induction to $b_1, \ldots, b_i', \ldots, b_n$ to obtain that $xb_1 \ldots b_i' \ldots b_n \in \mathrm{SN}$, hence $xb_1 \ldots b_i \ldots b_n \in \mathrm{SN}$. ∎

LEMMA 3.6 (intersection). $(I \neq \varnothing$ *and* $\forall i \in I.\, S_i \in \mathrm{SAT}) \Rightarrow (\bigcap_{i \in I} S_i) \in \mathrm{SAT}$.

Proof. Sat_0: $\bigcap_{i \in I} S_i \subseteq \mathrm{SN}$: let j be an element of I: $\bigcap_{i \in I} S_i \subseteq S_j \subseteq \mathrm{SN}$.

Sat_1: $a \in \mathrm{SN}, b[x \leftarrow a]\, b_1 \ldots b_n \in \bigcap_{i \in I} S_i \Rightarrow (\lambda x.b)\, ab_1 \ldots b_n \in \bigcap_{i \in I} S_i$:

let $a \in \mathrm{SN}, b[x \leftarrow a]\, b_1 \ldots b_n \in \bigcap_{i \in I} S_i$; by def. of $\bigcap$: $\forall i \in I. b[x \leftarrow a]\, b_1 \ldots b_n \in S_i$
by Sat_1 of S_i: $\forall i \in I.\ (\lambda x.b)\, ab_1 \ldots b_n \in S_i$
by def. of $\bigcap$: $(\lambda x.b)\, ab_1 \ldots b_n \in \bigcap_{i \in I} S_i$.

Sat_2: $b_1, ..., b_n \in \mathrm{SN} \Rightarrow x b_1 ... b_n \in \bigcap_{i \in I} s_i$;
let $b_1, ..., b_n \in \mathrm{SN}$; by Sat_2 on S_i: $\forall i \in I.\ b[x \leftarrow a]\ b_1 ... b_n \in S_i$
by def. of $\bigcap$: $b[x \leftarrow a]\ b_1 ... b_n \in \bigcap_{i \in I} S_i$. ∎

Notation 3.7 ($\mathrm{Min}_{\mathrm{SAT}}$). $\mathrm{Min}_{\mathrm{SAT}} = \bigcap_{\iota \in \mathrm{SAT}} \iota$.

Remark 3.8 ($\mathrm{Min}_{\mathrm{SAT}}$). $\mathrm{Min}_{\mathrm{SAT}}$ is a well-defined saturated set: it is well defined since, by Lemma 3.5, SAT is not empty. It is saturated by Lemma 3.6.

3.2. *The Theorem*

DEFINITION 3.9 (type erasure).

$$\begin{aligned}
\mathrm{typeErasure}(x) &= x \\
\mathrm{typeErasure}(\lambda x{:}A.a) &= \lambda x.\mathrm{typeErasure}(a) \\
\mathrm{typeErasure}(a(a')) &= \mathrm{typeErasure}(a)(\mathrm{typeErasure}(a')) \\
\mathrm{typeErasure}(\Lambda t \leqslant A.a) &= \mathrm{typeErasure}(a) \\
\mathrm{typeErasure}(a\{A\}) &= \mathrm{typeErasure}(a).
\end{aligned}$$

Notation 3.10 ($|\Gamma|, |\Delta|, |E|$). $|t_1 \leqslant A_1, ..., t_n \leqslant A_n| = |x_1{:}\ A_1, ..., x_n{:}\ A_n| = |t_1, ..., t_n| = n$.

Notation 3.11 (S^n). As usual, for any set S, we define $S^n = (...(S^0 \times S_{(1)}) \times ... \times S_{(n)})$, where S^0 is an arbitrary singleton $\{s\}$, to deal smoothly with the nullary case. Similarly, we define $\langle s_1, ..., s_n \rangle = \langle ...\langle s, s_1 \rangle, ..., s_n \rangle$.

Notation 3.12 (Δ^0, SAT^0). Δ^0 is an arbitrary singleton (e.g., $\{x\}$) used as a unit in products of subsets of Δ. Similarly, SAT^0 is an arbitrary singleton (e.g., $\{\mathrm{Min}_{\mathrm{SAT}}\}$) used as a unit for products of subsets of SAT.

Notation 3.13 ($a_{\Delta \leftarrow \delta}$). If Δ is a value environment $x_1{:}\ A_1, ..., x_n{:}\ A_n$ and δ is a tuple $\langle a_1, ..., a_n \rangle \subseteq \Delta^n$, then $\Delta \leftarrow \delta$ is the substitution $[x_1 \leftarrow a_1, ..., x_n \leftarrow a_n]$, and $a_{\Delta \leftarrow \delta}$ is the result of applying $\Delta \leftarrow \delta$ to a.

LEMMA 3.14. *The type erasure of any F-bounded term is a terminating λ term.*

Proof. We define five "semantic functions" which interpret any provable type environment, value environment, type, subtype, or term formation judgement. We prove that the interpretation of each provable judgement satisfies an associated "semantic condition." These conditions imply $\beta\eta$ strong normalization for any type-erased F-bounded term.

Informally, a type is interpreted by a set of λ-terms, a type environment $t_1 \leqslant T_1, ..., t_n \leqslant T_n$ by a set of n-tuples of sets, where each n-tuple specifies a possible way of associating a set with each type variable, a value environment is interpreted by a set of tuples of λ-terms, where each tuple specifies a "well-typed" assignment of λ-terms to the value variables, a value is interpreted by its type erasure, with a well-typed substitution applied to its free value variables.

The semantic conditions specify that any type is a saturated set, each term belongs to its type, no environment interpretation may be empty (empty environments would make the other soundness conditions useless, since those conditions are quantified on variables ranging over environment interpretations).

Here are the interpretation and the conditions:

$$
\begin{array}{rlll}
([\![]\!]\varnothing\text{TypeEnv}) & \text{Def.:} & [\![() \vdash \diamond]\!] & = \text{SAT}^0\ (=\{\text{Min}_{\text{SAT}}\}) \\
([\![]\!] \neq \varnothing\text{TypeEnv}) & \text{Def.:} & [\![\Gamma, t \leqslant A \vdash \diamond]\!] & = \{\langle \gamma, \iota\rangle \mid \gamma \in [\![\Gamma \vdash \diamond]\!], \iota \in \text{SAT}, \\
 & & & \quad \iota \subseteq [\![\text{vars}(\Gamma), t \vdash A]\!]\langle \gamma, \iota\rangle\} \\
([\![]\!]\text{TypeEnv}) & \text{Cond.:} & [\![\Gamma \vdash \diamond]\!] \subseteq \text{SAT}^{|\Gamma|} \text{ and } [\![\Gamma \vdash \diamond]\!] \neq \varnothing & \\
([\![]\!]\text{ValueEnv}) & \text{Def.:} & \forall \gamma \in [\![\Gamma \vdash \diamond]\!].\, [\![\Gamma, x_1 : A_1, \ldots, x_n : A_n \vdash \diamond]\!]\ \gamma & \\
 & & \quad = \Lambda^0 \times [\![\text{vars}(\Gamma) \vdash A_1]\!]\ \gamma \times \ldots \times [\![\text{vars}(\Gamma) \vdash A_n]\!]\gamma & \\
([\![]\!]\text{ValueEnv}) & \text{Cond.:} & \forall \gamma \in [\![\Gamma \vdash \diamond]\!].\, [\![\Gamma, x_1 : A_1, \ldots, x_n : A_n \vdash \diamond]\!]\ \gamma \neq \varnothing & \\
([\![]\!]\text{Var}) & \text{Def.:} & \forall \langle \iota_1, \ldots, \iota_n\rangle \in \text{SAT}^n.\, [\![t_1, \ldots, t_n \vdash t_i]\!]\langle \iota_1, \ldots, \iota_n\rangle = \iota_i & \\
([\![]\!]\text{Top}) & \text{Def.:} & \forall \gamma \in \text{SAT}^{|E|}.\, [\![E \vdash \text{Top}]\!]\ \gamma = \text{SN} & \\
([\![]\!] \to) & \text{Def.:} & \forall \gamma \in \text{SAT}^{|E|}.\, [\![E \vdash A \to B]\!]\ \gamma & \\
 & & \quad = \{b \in \Lambda \mid a \in [\![E \vdash A]\!]\ \gamma \Rightarrow b(a) \in [\![E \vdash B]\!]\ \gamma\} & \\
([\![]\!]\forall) & \text{Def.:} & \forall \gamma \in \text{SAT}^{|E|}.\, [\![E \vdash \forall t \leqslant A.B]\!]\ \gamma & \\
 & & \quad = \bigcap_{\iota \in \text{SAT},\, \iota \subseteq [\![E,\, t \vdash A]\!]\langle \gamma,\, \iota\rangle} [\![E, t \vdash B]\!]\langle \gamma, \iota\rangle & \\
([\![]\!]\text{Type}) & \text{Cond.:} & \forall \gamma \in \text{SAT}^{|E|}.\, [\![E \vdash A]\!]\ \gamma \in \text{SAT} & \\
([\![]\!] \leqslant) & \text{Def.:} & \forall \gamma \in [\![\Gamma \vdash \diamond]\!].\, [\![\Gamma \vdash A \leqslant B]\!]\ \gamma = \langle [\![\text{vars}(\Gamma) \vdash A]\!]\ \gamma, & \\
 & & [\![\text{vars}(\Gamma) \vdash B]\!]\ \gamma\rangle & \\
([\![]\!] \leqslant) & \text{Cond.:} & \forall \gamma \in [\![\Gamma \vdash \diamond]\!].\, \pi_1 [\![\Gamma \vdash A \leqslant B]\!]\ \gamma \subseteq \pi_2 [\![\Gamma \vdash A \leqslant B]\!]\gamma & \\
([\![]\!]\text{Term}) & \text{Def.:} & \forall \gamma \in [\![\Gamma \vdash \diamond]\!].\, \forall \delta \in [\![\Gamma, \Delta \vdash \diamond]\!]\ \gamma.\, [\![\Gamma, \Delta \vdash a : A]\!]\ \gamma\delta & \\
 & & \quad = \text{typeErasure}(a)_{\Delta \leftarrow \delta} & \\
([\![]\!]\text{Term}) & \text{Cond.:} & \forall \gamma \in [\![\Gamma \vdash \diamond]\!].\, \forall \delta \in [\![\Gamma, \Delta \vdash \diamond]\!]\gamma.\, [\![\Gamma, \Delta \vdash a : A]\!]\ \gamma\delta \in & \\
 & & [\![\text{vars}(\Gamma) \vdash A]\!]\gamma. &
\end{array}
$$

We prove that the interpretation of any provable judgement satisfies the associated condition by induction on its proof tree by showing that, for each rule, if the interpretation of the premises is sound, the interpretation of the consequences is sound too. This proof will be carried out in the next five sections.

Assuming that the soundness of the interpretation will be proved, we can now prove the lemma. Let a be an F-bounded term typed in an environment Γ, Δ. Take any $\gamma_0 \in [\![\Gamma \vdash \diamond]\!]$ (one exists, by condition ($[\![]\!]$TypeEnv)), and observe that the tuple of λ-terms vars(Δ) belongs to $[\![\Gamma, \Delta \vdash \diamond]\!]\gamma_0$ since any variable belongs to any saturated set. By condition ($[\![]\!]$Term), $[\![\Gamma, \Delta \vdash a : A]\!]\ \gamma_0\ \text{vars}(\Delta)$ ($=\text{typeErasure}(a)$) $\in [\![\text{vars}(\Gamma) \vdash A]\!]\gamma_0$. By condition ($[\![]\!]$Type), $[\![\text{vars}(\Gamma) \vdash A]\!]\gamma_0$ is saturated hence, by Sat_0, typeErasure(a) is strongly normalizing. ∎

THEOREM 3.15. *F-bounded is strongly normalizing.*

Proof. Consider a $\beta - \eta - \beta 2 - \eta 2$ reduction chain R starting from an F-bounded term and the chain typeErasure(R) consisting of all the type erasures of the elements of R. A $\beta - \eta$ step corresponds in typeErasure(R) to each $\beta - \eta$ step in R, while two identical terms correspond in typeErasure(R) to each pair of terms related by a $\beta 2 - \eta 2$ step in R. Hence, if we collapse all sequences of identical terms in typeErasure(R), we still obtain a $\beta - \eta$ reduction chain "collapse(typeErasure(R))" in Λ. Since each $\beta 2 - \eta 2$ step strictly reduces the number of Λ symbols in the term, any sequence of $\beta 2 - \eta 2$ reductions in R has a finite length. Hence, if R were

infinite, then collapse(typeErasure(R)) would be infinite too. Since collapse(type Erasure(R)) is finite by the previous Lemma, then R is finite too. ▮

In the next sections we prove that each rule, when applied to judgements with a sound interpretation, yields a judgement with a sound interpretation, thus completing the proof of the theorem.

3.3. *Soundness of Type Environment Rules*

Interpretation.

($[\![]\!]\varnothing$TypeEnv)
$$[\![() \vdash \diamond]\!] = \mathrm{SAT}^0(\{\mathrm{Min}_{\mathrm{SAT}}\})$$

($[\![]\!] \neq \varnothing$TypeEnv)
$$[\![\Gamma, t \leqslant A \vdash \diamond]\!] = \{\langle \gamma, \iota\rangle \mid \gamma \in [\![\Gamma \vdash \diamond]\!], \iota \in \mathrm{SAT}, \iota \subseteq [\![\mathrm{vars}(\Gamma), t \vdash A]\!]\langle \gamma, \iota\rangle\}.$$

In $F_{\leqslant}$, an environment $\Gamma, t \leqslant A$ would be interpreted as $\{\langle \gamma, \iota\rangle \mid \gamma \in [\![\Gamma \vdash \diamond]\!], \iota \subseteq [\![\Gamma \vdash A]\!]\gamma\}$ [Bruce Longo 90]. In F-bounded, this cannot be generalized as $\{\langle \gamma, \iota\rangle \mid \gamma \in [\![\Gamma \vdash \diamond]\!], \iota \subseteq [\![\Gamma, t \leqslant A \vdash A]\!]\langle \gamma, \iota\rangle\}$, because $[\![\Gamma, t \leqslant A \vdash A]\!]$ is, in turn, defined in terms of $[\![\Gamma, t \leqslant A \vdash \diamond]\!]$. We avoided this circularity in the rules by defining the notion of *name environment*, and the same technique is used here to give a well-founded interpretation for type environments. Note that no circularity is hidden in the condition $\iota \subseteq [\![\mathrm{vars}(\Gamma), t \vdash A]\!]\langle \gamma, \iota\rangle$, which is just a set inclusion with ι appearing on both sides. $[\![\mathrm{vars}(\Gamma), t \vdash A]\!]\langle \gamma, \iota\rangle$ is well defined since $\gamma \in [\![\Gamma \vdash \diamond]\!] \subseteq (\mathrm{SAT}(A))^{|\Gamma|}$ by induction, and $\iota \in \mathrm{SAT}$ by construction.

Condition. ($[\![]\!]$TypeEnv) $[\![\Gamma \vdash \diamond]\!] \subseteq \mathrm{SAT}^{|\Gamma|}(a)$ and $[\![\Gamma \vdash \diamond]\!] \neq \varnothing(b)$.

($[\![]\!]\varnothing$TypeEnv): $\mathrm{SAT}^0 \subseteq \mathrm{SAT}^0$ (a) and $\mathrm{SAT}^0 \neq \varnothing$ (b) are both true by definition.

($[\![]\!] \neq \varnothing$TypeEnv): The rule is: $\Gamma \vdash \diamond, \mathrm{vars}(\Gamma), t \vdash A, t \notin \Gamma \Rightarrow \Gamma, t \leqslant A \vdash \diamond$

Soundness. Hyp.: $[\![\Gamma \vdash \diamond]\!] \subseteq \mathrm{SAT}^{|\Gamma|}$, $[\![\Gamma \vdash \diamond]\!] \neq \varnothing$,
$\forall \langle \gamma, \iota\rangle \in \mathrm{SAT}^{|\Gamma|+1}. [\![\mathrm{vars}(\Gamma), t \vdash A]\!]\langle \gamma, \iota\rangle \in \mathrm{SAT}$

Th.: $[\![\Gamma, t \leqslant A \vdash \diamond]\!] \subseteq \mathrm{SAT}^{|\Gamma|+1}$ (a)

$[\![\Gamma, t \leqslant A \vdash \diamond]\!] \neq \varnothing$ (b)

Proof. (a) $[\![\Gamma, t \leqslant A \vdash \diamond]\!] = \{\langle \gamma, \iota\rangle \mid \gamma \in [\![\Gamma \vdash \diamond]\!], \iota \in \mathrm{SAT}, \ldots\}$ is a set of pairs $\gamma \in \mathrm{SAT}^{|\Gamma|}$ and $\iota \in \mathrm{SAT}$, hence is included in $\mathrm{SAT}^{|\Gamma|+1}$.

(b) $[\![\Gamma, t \leqslant A \vdash \diamond]\!] \neq \varnothing$: By Hyp., $\exists \gamma_0 \in [\![\Gamma \vdash \diamond]\!]$. Then, $\langle \gamma_0, \mathrm{Min}_{\mathrm{SAT}}\rangle \in [\![\Gamma, t \leqslant A \vdash \diamond]\!]$, since $\mathrm{Min}_{\mathrm{SAT}}$, the minimal saturated set, is included in $[\![\mathrm{vars}(\Gamma), t \vdash A]\!]\langle \gamma_0, \mathrm{Min}_{\mathrm{SAT}}\rangle$, which is saturated by Hyp. ▮

3.4. *Soundness of Value Environment Judgments*

Interpretation.

($[\![]\!]$ValueEnv)
$$[\![\Gamma, x_1 : A_1, \ldots, x_n : A_n \vdash \diamond]\!]\gamma = A^0 \times [\![\mathrm{vars}(\Gamma) \vdash A_1]\!]\gamma \times \ldots \times [\![\mathrm{vars}(\Gamma) \vdash A_n]\!]\gamma.$$

Value environments can be interpreted by a plain cartesian product, since the types in a value environment do not depend on the previous value variables.

Condition. ($[\![]\!]$ValueEnv) $\forall \gamma \in [\![\Gamma \vdash \diamond]\!]$. $A^0 \times [\![\mathrm{vars}(\Gamma) \vdash A_1]\!]\gamma \times \ldots \times [\![\mathrm{vars}(\Gamma) \vdash A_n]\!]\gamma \neq \varnothing$.

Proof. The product is not empty since every saturated set contains at least every variable. ▮

3.5. *Soundness of Typing Rules*

Interpretation.
$$(\llbracket\rrbracket\text{Var}) \quad \forall \langle \iota_1, ..., \iota_n \rangle \in \text{SAT}^n . \llbracket t_1, ..., t_n \vdash t_i \rrbracket \langle \iota_1, ..., \iota_n \rangle = \iota_i$$
$$(\llbracket\rrbracket\text{Top}) \quad \forall \gamma \in \text{SAT}^{|E|} . \llbracket E \vdash \text{Top} \rrbracket \gamma = \text{SN}$$
$$(\llbracket\rrbracket \to) \quad \forall \gamma \in \text{SAT}^{|E|} . \llbracket E \vdash A \to B \rrbracket \gamma = \{ b \in \Lambda \mid a \in \llbracket E \vdash A \rrbracket \gamma \Rightarrow b(a) \in \llbracket E \vdash B \rrbracket \gamma \}$$
$$(\llbracket\rrbracket\forall) \quad \forall \gamma \in \text{SAT}^{|E|} . \llbracket E \vdash \forall t \leqslant A . B \rrbracket \gamma = \bigcap_{\iota \in \text{SAT}, \iota \subseteq \llbracket E, t \vdash A \rrbracket \langle \gamma, \iota \rangle} \llbracket E, t \vdash B \rrbracket \langle \gamma, \iota \rangle .$$

Type variables are interpreted by the type environment. Top is the set of all strongly normalizing λ-terms. A functional type $\llbracket A \to B \rrbracket$ contains all terms which, applied to a term in $\llbracket A \rrbracket$, yield a term in $\llbracket B \rrbracket$; note that we mean a bare syntactic application with no evaluation. Quantification is interpreted by intersection.

Condition. $(\llbracket\rrbracket\text{Type}) \, \forall \gamma \in \text{SAT}^{|E|} . \llbracket E \vdash A \rrbracket \gamma \in \text{SAT}.$

(Var Form) $E, t, E' \vdash \diamond \Rightarrow E, t, E' \vdash t$

$\forall \langle \iota_1, ..., \iota_n \rangle \in \text{SAT}^n . \llbracket t_1, ..., t_n \vdash t_i \rrbracket \langle \iota_1, ..., \iota_n \rangle = \iota_i$ belongs to SAT by construction.

(Top Form) $E, t, E' \vdash \diamond \Rightarrow E, t, E' \vdash \text{Top}$

$\forall \gamma \in \text{SAT}^{|E|} . \llbracket E \vdash \text{Top} \rrbracket \gamma = \text{SN}$ belongs to SAT by Lemma 3.5 (SN).

($\to$ Form) $E \vdash A, E \vdash B \Rightarrow E \vdash A \to B$

Soundness. Hyp.: $\forall \gamma \in \text{SAT}^{|E|} . \llbracket E \vdash A \rrbracket \gamma \in \text{SAT}$ and $\llbracket E \vdash B \rrbracket \gamma \in \text{SAT}$.
Th.: $\forall \gamma \in \text{SAT}^{|E|} . \llbracket E \vdash A \to B \rrbracket \gamma \in \text{SAT}$.

Sat_0: $\llbracket E \vdash A \to B \rrbracket \gamma \subseteq \text{SN}$.

Let $f \in \llbracket E \vdash A \to B \rrbracket \gamma$. Consider any variable x; $x \in \llbracket E \vdash A \rrbracket \gamma$ by Sat_2, hence $f(x) \in \llbracket E \vdash B \rrbracket \gamma$ by definition of $\llbracket E \vdash A \to B \rrbracket \gamma$. $f(x) \in \text{SN}$ by Sat_0, hence $f \in \text{SN}$.

Sat_1: $a \in \text{SN}, b[x \backslash a] \, b_1 ... b_n) \in \llbracket E \vdash A \to B \rrbracket \gamma \Rightarrow (\lambda x . b) \, a b_1 ... b_n \in \llbracket E \vdash A \to B \rrbracket \gamma$

Let:	$a \in \text{SN}, (b[x \backslash a] \, b_1 ... b_n) \in \llbracket E \vdash A \to B \rrbracket \gamma$
By def. of $\llbracket E \vdash A \to B \rrbracket \gamma$:	$\forall a' \in \llbracket E \vdash A \rrbracket \gamma . (b[x \backslash a] \, b_1 ... b_n) \, a' \in \llbracket E \vdash B \rrbracket \gamma$
By Sat_1 for $\llbracket E \vdash B \rrbracket \gamma$:	$\forall a' \in \llbracket E \vdash A \rrbracket \gamma . ((\lambda x . b) \, a b_1 ... b_n) \, a' \in \llbracket E \vdash B \rrbracket \gamma$
By def. of $\llbracket E \vdash A \to B \rrbracket \gamma$:	$(\lambda x . b) \, a b_1 ... b_n \in \llbracket E \vdash A \to B \rrbracket \gamma$.

Sat_2: $b_1, ..., b_n \in \text{SN} \Rightarrow x b_1 ... b_n \in \llbracket E \vdash A \to B \rrbracket \gamma$

Let:	$b_1, ..., b_n \in \text{SN}$
By Sat_2 for $\llbracket E \vdash B \rrbracket \gamma$:	$\forall a \in \text{SN} . x b_1 ... b_n a \in \llbracket E \vdash B \rrbracket \gamma$
By $\llbracket E \vdash A \rrbracket \gamma \subseteq \text{SN}$:	$\forall a \in \llbracket E \vdash A \rrbracket \gamma . x b_1 ... b_n a \in \llbracket E \vdash B \rrbracket \gamma$
By def. of $\llbracket E \vdash A \to B \rrbracket \gamma$:	$x b_1 ... b_n \in \llbracket E \vdash A \to B \rrbracket \gamma$.

($\forall$Form) $E, t \vdash A, \quad E, t \vdash B \Rightarrow E \vdash \forall t \leqslant A . B.$

Soundness. Hyp.: $\forall \langle \gamma, \iota \rangle \in \mathrm{SAT}^{|E|+1}. [\![E, t \vdash A]\!] \langle \gamma, \iota \rangle \in \mathrm{SAT}$ and $[\![E, t \vdash B]\!] \langle \gamma, \iota \rangle \in \mathrm{SAT}$

Th.: $\forall \gamma \in \mathrm{SAT}^{|E|}. [\![E \vdash \forall t \leqslant A.B]\!] \gamma =_{\mathrm{def}} \bigcap_{\iota \in \mathrm{SAT}, \iota \subseteq [\![E, t \vdash A]\!] \langle \gamma, \iota \rangle} [\![E, t \vdash B]\!] \langle \gamma, \iota \rangle \in \mathrm{SAT}.$

Proof. $\{\iota \mid \iota \in \mathrm{SAT}, \iota \subseteq [\![E, t \vdash A]\!] \langle \gamma, \iota \rangle\}$ is not empty since it contains, at least, $\mathrm{Min}_{\mathrm{SAT}}(\mathrm{Min}_{\mathrm{SAT}} \subseteq [\![E, t \vdash A]\!] \langle \gamma, \mathrm{Min}_{\mathrm{SAT}} \rangle)$. We can now apply Lemma 3.6 (Intersection), by observing that each $[\![E, t \vdash B]\!] \langle \gamma, \iota \rangle$ is saturated by Hyp., to conclude that $[\![E \vdash \forall t \leqslant A.B]\!] \gamma \in \mathrm{SAT}$. ∎

We now present some lemmas about type interpretation which will be used in the next sections.

LEMMA 3.16 (Weakening). *If* $E, t, E' \vdash A$ *and* $E, E' \vdash A$ *hold, then*:

$$\forall \gamma \in \mathrm{SAT}^{|E|}, \iota \in \mathrm{SAT}, \gamma' \in \mathrm{SAT}^{|E'|}. [\![E, t, E' \vdash A]\!] \langle \gamma, \iota, \gamma' \rangle = [\![E, E' \vdash A]\!] \langle \gamma, \gamma' \rangle.$$

Proof. By induction and by cases on the shape of A. Cases $A = u$ (with $u \neq t$) and $A = \mathrm{Top}$ are immediate. $A = t$ is excluded by the hypothesis. Cases $A = A' \to A''$ and $A = \forall u \leqslant A'.A''$ are immediate by induction. ∎

LEMMA 3.17 (ValueEnvWeakening). *If* $\Gamma, t \leqslant A, \Gamma', \Delta \vdash \diamond$ *and* $\Gamma, \Gamma', \Delta \vdash \diamond$ *hold, then*:

$$\forall \gamma \in \mathrm{SAT}^{|E|}, \iota \in \mathrm{SAT}, \gamma' \in \mathrm{SAT}^{|E'|}. [\![\Gamma, t \leqslant A, \Gamma', \Delta \vdash \diamond]\!] \langle \gamma, \iota, \gamma' \rangle = [\![\Gamma, \Gamma', \Delta \vdash \diamond]\!] \langle \gamma, \gamma' \rangle.$$

Proof. It is a corollary of Lemma 3.16 (Weakening), since:

$$[\![\Gamma, t \leqslant A, \Gamma', x_1 : A_1, ..., x_n : A_n \vdash \diamond]\!] \gamma =_{\mathrm{def}} A^0 \times [\![\mathrm{vars}(\Gamma), t, \mathrm{vars}(\Gamma') \vdash A_1]\!] \gamma \times ... \times [\![\mathrm{vars}(\Gamma), t, \mathrm{vars}(\Gamma') \vdash A_n]\!] \gamma.$$ ∎

LEMMA 3.18 (TypeSubst). *For any type formation judgment* $\Gamma'' \vdash B[t \leftarrow A]$, *for any way of splitting* Γ'' *into two parts* Γ, Γ' *such that* $[\![\mathrm{vars}(\Gamma) \vdash A]\!]$; *i.e., such that* $\forall t' \in \mathrm{vars}(\Gamma').t' \notin \mathrm{FTV}(A)$:

$$\forall \gamma \in \mathrm{SAT}^{|\Gamma|}, \gamma' \in \mathrm{SAT}^{|\Gamma|}. [\![\mathrm{vars}(\Gamma), \mathrm{vars}(\Gamma') \vdash B[t \leftarrow A]]\!] \langle \gamma, \gamma' \rangle = [\![\mathrm{vars}(\Gamma), t, \mathrm{vars}(\Gamma') \vdash B]\!] \langle \gamma, [\![\mathrm{vars}(\Gamma) \vdash A]\!] \gamma, \gamma' \rangle.$$

Proof. By induction and by cases on the shape of B. Cases $B = u$ (with $u \neq t$) and $B = \mathrm{Top}$ are corollaries of Lemma 3.16 (Weakening).

$B = t$: $[\![\mathrm{vars}(\Gamma), \mathrm{vars}(\Gamma') \vdash t[t \leftarrow A]]\!]\langle \gamma, \gamma' \rangle$
$= [\![\mathrm{vars}(\Gamma), \mathrm{vars}(\Gamma') \vdash A]\!]\langle \gamma, \gamma' \rangle$
By Lemma 3.16 (Weakening): $= [\![\mathrm{vars}(\Gamma) \vdash A]\!]\langle \gamma \rangle$
By def. ($[\![]\!]$Var): $= [\![\mathrm{vars}(\Gamma), t, \mathrm{vars}(\Gamma') \vdash t]\!]\langle \gamma, [\![\mathrm{vars}(\Gamma) \vdash A]\!]\ \gamma, \gamma' \rangle$.

$B = \forall t' \leqslant B'.B''$: $[\![\mathrm{vars}(\Gamma), \mathrm{vars}(\Gamma') \vdash (\forall t' \leqslant B'.B'')[t \leftarrow A]]\!]\langle \gamma, \gamma' \rangle$
By ($[\![]\!]\forall$): $= \bigcap_{\iota \in \mathrm{SAT},\ \iota \subseteq [\![\mathrm{vars}(\Gamma), \mathrm{vars}(\Gamma'), t' \vdash B'[t \leftarrow A]]\!]\langle \gamma, \gamma', \iota \rangle}$
$[\![\mathrm{vars}(\Gamma), \mathrm{vars}(\Gamma'), t' \vdash B''[t \leftarrow A]]\!]\langle \gamma, \gamma', \iota \rangle$
By ind.: $= \bigcap_{\iota \in \mathrm{SAT},\ \iota \subseteq [\![\mathrm{vars}(\Gamma), t, \mathrm{vars}(\Gamma'), t' \vdash B']\!]\langle \gamma, [\![\mathrm{vars}(\Gamma) \vdash A]\!]\ \gamma, \gamma', \iota \rangle}$
$[\![\mathrm{vars}(\Gamma), t, \mathrm{vars}(\Gamma'), t' \vdash B'']\!]\langle \gamma, [\![\mathrm{vars}(\Gamma) \vdash A]\!]\ \gamma, \gamma', \iota \rangle$
By ($[\![]\!]\forall$): $= [\![\mathrm{vars}(\Gamma), t, \mathrm{vars}(\Gamma') \vdash \forall t' \leqslant B'.B'']\!]\langle \gamma, [\![\mathrm{vars}(\Gamma) \vdash A]\!]\ \gamma, \gamma' \rangle$.

$B = B' \to B''$: similar but easier. ∎

3.6. *Soundness of Subtype Rules*

Interpretation.
($[\![]\!] \leqslant$)$\forall \gamma \in [\![\Gamma \vdash \diamond]\!].[\![\Gamma \vdash A \leqslant B]\!]\,\gamma = \langle [\![\mathrm{vars}(\Gamma) \vdash A]\!]\ \gamma,\ [\![\mathrm{vars}(\Gamma) \vdash B]\!]\,\gamma \rangle$.

Condition. ($[\![]\!] \leqslant$) $\forall \gamma \in [\![\Gamma \vdash \diamond]\!].\pi_1 [\![\Gamma \vdash A \leqslant B]\!]\ \gamma \subseteq \pi_2 [\![\Gamma \vdash A \leqslant B]\!]\,\gamma$;
i.e., $\forall \gamma \in [\![\Gamma \vdash \diamond]\!].[\![\mathrm{vars}(\Gamma) \vdash A]\!]\ \gamma \subseteq [\![\mathrm{vars}(\Gamma) \vdash B]\!]\,\gamma$

(Var $\leqslant$) $\Gamma, t \leqslant A, \Gamma' \vdash \diamond \Rightarrow \Gamma, t \leqslant A, \Gamma' \vdash t \leqslant A$

Soundness. $\forall \gamma \in \mathrm{SAT}^{|\Gamma|}, \iota \in \mathrm{SAT}, \gamma' \in \mathrm{SAT}^{|\Gamma|}.\langle \gamma, \iota, \gamma' \rangle \in [\![\Gamma, t \leqslant A, \Gamma' \vdash \diamond]\!]$
$\Rightarrow [\![\mathrm{vars}(\Gamma), t, \mathrm{vars}(\Gamma') \vdash t]\!]\langle \gamma, \iota, \gamma' \rangle$
$\subseteq [\![\mathrm{vars}(\Gamma), t, \mathrm{vars}(\Gamma') \vdash A]\!]\langle \gamma, \iota, \gamma' \rangle$

Let:	$\langle \gamma, \iota, \gamma' \rangle \in [\![\Gamma, t \leqslant A, \Gamma' \vdash \diamond]\!]$	(a)
By definition of $[\![\Gamma \vdash \diamond]\!]$:	$\langle \gamma, \iota \rangle \in [\![\Gamma, t \leqslant A \vdash \diamond]\!]$	(b)
By the same definition:	$\iota \subseteq [\![\mathrm{vars}(\Gamma), t \vdash A]\!]\langle \gamma, \iota \rangle$	(c)
By definition ($[\![]\!]$Var):	$\iota = [\![\mathrm{vars}(\Gamma), t, \mathrm{vars}(\Gamma') \vdash t]\!]\langle \gamma, \iota, \gamma' \rangle$	(d)
By Lemma 3.16 (Weakening):	$[\![\mathrm{vars}(\Gamma), t \vdash A]\!]\langle \gamma, \iota \rangle$	
	$= [\![\mathrm{vars}(\Gamma), t, \mathrm{vars}(\Gamma') \vdash A]\!]\langle \gamma, \iota, \gamma' \rangle$	(e)
Substituting (d) and (e) in (c):	$[\![\mathrm{vars}(\Gamma), t, \mathrm{vars}(\Gamma') \vdash t]\!]\langle \gamma, \iota, \gamma' \rangle$	
	$\subseteq [\![\mathrm{vars}(\Gamma), t, \mathrm{vars}(\Gamma') \vdash A]\!]\langle \gamma, \iota, \gamma' \rangle$.	

(Top $\leqslant$) $\Gamma \vdash \diamond, \mathrm{vars}(\Gamma) \vdash A \Rightarrow \Gamma \vdash A \leqslant \mathrm{Top}$.

Soundness. Hyp.: $[\![\Gamma \vdash]\!] \subseteq \mathrm{SAT}^{|\Gamma|}$, $[\![\Gamma \vdash]\!] \neq \varnothing$,
$\forall \gamma \in \mathrm{SAT}^{|\Gamma|}.[\![\mathrm{vars}(\Gamma) \vdash A]\!]\ \gamma \in \mathrm{SAT}$
Th.: $\forall \gamma \in [\![\Gamma \vdash]\!].[\![\mathrm{vars}(\Gamma) \vdash A]\!]\ \gamma \subseteq [\![\mathrm{vars}(\Gamma) \vdash \mathrm{Top}]\!]\ \gamma = \mathrm{SN}$

By Hyp., $[\![\mathrm{vars}(\Gamma) \vdash A]\!]\ \gamma \in \mathrm{SAT}$; hence, $[\![\mathrm{vars}(\Gamma) \vdash A]\!]\ \gamma \subseteq \mathrm{SN}$ by Sat_0.

$(\to \leqslant)$ $\Gamma \vdash A \leqslant A', \Gamma \vdash B \leqslant B' \Rightarrow \Gamma \vdash A' \to B \leqslant A \to B'$.

Soundness. Hyp.: $\forall \gamma \in [\![\Gamma \vdash]\!] . [\![\mathrm{vars}(\Gamma) \vdash A]\!] \gamma \subseteq [\![\mathrm{vars}(\Gamma) \vdash A']\!] \gamma$, (a)
$\forall \gamma \in [\![\Gamma \vdash]\!] . [\![\mathrm{vars}(\Gamma) \vdash B]\!] \gamma \subseteq [\![\mathrm{vars}(\Gamma) \vdash B']\!] \gamma$ (b)
Th.: $\forall \gamma \in [\![\Gamma \vdash]\!] . \forall f \in [\![\mathrm{vars}(\Gamma) \vdash A' \to B]\!] \gamma .$
$f \in [\![\mathrm{vars}(\Gamma) \vdash A \to B']\!] \gamma$.

Let: $\gamma \in [\![\Gamma \vdash]\!], f \in [\![\mathrm{vars}(\Gamma) \vdash A' \to B]\!] \gamma$
By def. ($[\![]\!] \to$): $\forall a \in [\![\mathrm{vars}(\Gamma) \vdash A']\!] \gamma . f(a) \in [\![\mathrm{vars}(\Gamma) \vdash B]\!] \gamma$
By (a): $\forall a \in [\![\mathrm{vars}(\Gamma) \vdash A]\!] \gamma . f(a) \in [\![\mathrm{vars}(\Gamma) \vdash B]\!] \gamma$
By (b): $\forall a \in [\![\mathrm{vars}(\Gamma) \vdash A]\!] \gamma . f(a) \in [\![\mathrm{vars}(\Gamma) \vdash B']\!] \gamma$
By def. ($[\![]\!] \to$): $f \in [\![\mathrm{vars}(\Gamma) \vdash A \to B']\!] \gamma$.

$(\forall \leqslant)$ $\Gamma, t \leqslant A \vdash t \leqslant A', \quad \Gamma, t \leqslant A \vdash B \leqslant B' \Rightarrow \Gamma \vdash \forall t \leqslant A' . B \leqslant \forall t \leqslant A . B'$.

Soundness. Hyp.: $\forall \langle \gamma, \iota \rangle \in [\![\Gamma, t \leqslant A \vdash \Diamond]\!] .$
$[\![\mathrm{vars}(\Gamma), t \vdash t]\!] \langle \gamma, \iota \rangle \subseteq [\![\mathrm{vars}(\Gamma), t \vdash A']\!] \langle \gamma, \iota \rangle$
$\forall \langle \gamma, \iota \rangle \in [\![\Gamma, t \leqslant A \vdash \Diamond]\!] .$
$[\![\mathrm{vars}(\Gamma), t \vdash B]\!] \langle \gamma, \iota \rangle \subseteq [\![\mathrm{vars}(\Gamma), t \vdash B']\!] \langle \gamma, \iota \rangle$
Th.: $\forall \gamma \in [\![\Gamma \vdash]\!] . \forall f \in \Lambda . f \in [\![\mathrm{vars}(\Gamma) \vdash \forall t \leqslant A' . B]\!] \gamma$
$\Rightarrow f \in [\![\mathrm{vars}(\Gamma) \vdash \forall t \leqslant A . B']\!] \gamma$.

Applying definitions ($[\![]\!] \neq \varnothing$TypeEnv) and ($[\![]\!] \forall$) we can rewrite Hyp. and Th., respectively, as

Hyp.: $\forall \gamma \in [\![\Gamma \vdash]\!] . \forall \iota \in \mathrm{SAT} . \iota \subseteq [\![\mathrm{vars}(\Gamma), t \vdash A]\!] \langle \gamma, \iota \rangle$
$\Rightarrow [\![\mathrm{vars}(\Gamma), t \vdash t]\!] \langle \gamma, \iota \rangle \subseteq [\![\mathrm{vars}(\Gamma), t \vdash A']\!] \langle \gamma, \iota \rangle$ (a)
$\forall \gamma \in [\![\Gamma \vdash]\!] . \forall \iota \in \mathrm{SAT} . \iota \subseteq [\![\mathrm{vars}(\Gamma), t \vdash A]\!] \langle \gamma, \iota \rangle$
$\Rightarrow [\![\mathrm{vars}(\Gamma), t \vdash B]\!] \langle \gamma, \iota \rangle \subseteq [\![\mathrm{vars}(\Gamma), t \vdash B']\!] \langle \gamma, \iota \rangle$ (b)
Th.: $\forall \gamma \in [\![\Gamma \vdash]\!] . \forall f \in \Lambda .$
$(\forall \iota \in \mathrm{SAT} . \iota \subseteq [\![\mathrm{vars}(\Gamma), t \vdash A']\!] \langle \gamma, \iota \rangle \Rightarrow f \in [\![\mathrm{vars}(\Gamma), t \vdash B]\!] \langle \gamma, \iota \rangle)$ (c)
$\Rightarrow (\forall \iota \in \mathrm{SAT} . \iota \subseteq [\![\mathrm{vars}(\Gamma), t \vdash A]\!] \langle \gamma, \iota \rangle$
$\Rightarrow f \in [\![\mathrm{vars}(\Gamma), t \vdash B']\!] \langle \gamma, \iota \rangle)$. (d)

Assuming (a), (b), and (c), we prove that (d) holds.

Proof. Let: $f \in \Lambda, \gamma \in [\![\Gamma \vdash]\!], \iota \in \mathrm{SAT}$ (e)
Let: $\iota \subseteq [\![\mathrm{vars}(\Gamma), t \vdash A]\!] \langle \gamma, \iota \rangle$ (f)
By (e), (f), and (a): $[\![\mathrm{vars}(\Gamma), t \vdash t]\!] \langle \gamma, \iota \rangle \subseteq$
$[\![\mathrm{vars}(\Gamma), t \vdash A']\!] \langle \gamma, \iota \rangle$ (g)
By def. ($[\![]\!]$Var and (g): $\iota \subseteq [\![\mathrm{vars}(\Gamma), t \vdash A']\!] \langle \gamma, \iota \rangle$ (h)
By (e), (h), and (c): $f \in [\![\mathrm{vars}(\Gamma), t \vdash B]\!] \langle \gamma, \iota \rangle$ (l)
By (l), (e), and (f), and (b): $f \in [\![\mathrm{vars}(\Gamma), t \vdash B']\!] \langle \gamma, \iota \rangle$. ∎

Id and Trans subtyping: soundness of these rules follows from the reflexivity and transitivity of set inclusion.

$(\mathrm{Id} \leqslant)$ $\Gamma \vdash \Diamond, \mathrm{vars}(\Gamma) \vdash A \Rightarrow \Gamma \vdash A \leqslant A$

$(\mathrm{Trans} \leqslant)$ $\Gamma \vdash A \leqslant B, \Gamma \vdash B \leqslant C \Rightarrow \Gamma \vdash A \leqslant C$.

3.7. *Interpretation and Soundness of Term Judgments*

Interpretation. ($[\![]\!]$Term)
$\forall \gamma \in [\![\Gamma \vdash \diamond]\!]. \forall \delta \in [\![\Gamma, \Delta \vdash \diamond]\!]\ \gamma. [\![\Gamma, \Delta \vdash a: A]\!]\ \gamma\delta = \text{typeErasure}(a)_{\Delta \leftarrow \delta}$.

Condition. ($[\![]\!]$Term)
$\forall \gamma \in [\![\Gamma \vdash \diamond]\!]. \forall \delta \in [\![\Gamma, \Delta \vdash \diamond]\!]\ \gamma. \text{typeErasure}(a)_{\Delta \leftarrow \delta} \in [\![\text{vars}(\Gamma) \vdash A]\!]\gamma$

(Var) $\Gamma, x_1: A_1, ..., x_n: A_n \vdash \diamond \Rightarrow \Gamma, x_1: A_1, ..., x_n: A_n \vdash x_i: A_i$

Soundness. Hyp.: $\forall \gamma \in [\![\Gamma \vdash \diamond]\!], \forall \langle a_1, ..., a_n \rangle \in [\![\Gamma, x_1: A_1, ..., x_n: A_n \vdash \diamond]\!]\gamma.$
$a_i \in [\![\text{vars}(\Gamma) \vdash A_i]\!]\gamma$
Th.: $\forall \gamma \in [\![\Gamma \vdash \diamond]\!]. \forall \langle a_1, ..., a_n \rangle \in [\![\Gamma, x_1: A_1, ..., x_n: A_n \vdash \diamond]\!]\gamma.$
$\text{typeErasure}(x_i)[x_1 \leftarrow a_1, ..., x_n \leftarrow a_n] \in [\![\text{vars}(\Gamma) \vdash A_i]\!]\gamma.$

($\rightarrow$Intro) $\Gamma, \Delta, x: A \vdash b: B \Rightarrow \Gamma, \Delta \vdash \lambda x: A.b: A \rightarrow B$.

Here the first saturation condition is used.

Soundness. Hyp.: $\forall \gamma \in [\![\Gamma \vdash \diamond]\!], \forall \langle \delta, a \rangle \in [\![\Gamma, \Delta, x: A \vdash \diamond]\!]\gamma.$
$\text{typeErasure}(b)_{(\Delta, x: A) \leftarrow \langle \delta, a \rangle} \in [\![\text{vars}(\Gamma) \vdash B]\!]\gamma.$
Th.: $\forall \gamma \in [\![\Gamma \vdash \diamond]\!]. \forall \delta \in [\![\Gamma, \Delta \vdash \diamond]\!]\gamma.$
$\text{typeErasure}(\lambda x: A.b)_{\Delta \leftarrow \delta} \in [\![\text{vars}(\Gamma) \vdash A \rightarrow B]\!]\gamma.$

By def. ($[\![]\!] \rightarrow$), Th. may be rewritten as:

Th.: $\forall \gamma \in [\![\Gamma \vdash \diamond]\!]. \forall \delta \in [\![\Gamma, \Delta \vdash \diamond]\!]\ \gamma. \forall a \in [\![\text{vars}(\Gamma) \vdash A]\!]\ \gamma.$
$(\text{typeErasure}(\lambda x: A.b)_{\Delta \leftarrow \delta})(a) \in [\![\text{vars}(\Gamma) \vdash B]\!]\gamma.$

Proof.

Let:	$\gamma \in [\![\Gamma \vdash \diamond]\!], \delta \in [\![\Gamma, \Delta \vdash \diamond]\!]\ \gamma, a \in [\![\text{vars}(\Gamma) \vdash A]\!]\gamma$
By def. ($[\![]\!]$ValueEnv):	$\langle \delta, a \rangle \in [\![\Gamma, \Delta, x: A \vdash \diamond]\!]\gamma$
By Hyp.:	$\text{typeErasure}(b)_{(\Delta, x: A) \leftarrow \langle \delta, a \rangle} \in [\![\text{vars}(\Gamma) \vdash B]\!]\gamma$
Splitting the substitution:	$\text{typeErasure}(b)_{\Delta \leftarrow \delta}[x \leftarrow a] \in [\![\text{vars}(\Gamma) \vdash B]\!]\gamma$
By cond. Sat_1:	$(\lambda x.\text{typeErasure}(b)_{\Delta \leftarrow \delta})(a)$
	$= (\text{typeErasure}(\lambda x: A.b)_{\Delta \leftarrow \delta})(a)$
	$\in [\![\text{vars}(\Gamma) \vdash B]\!]\gamma.$ ∎

($\rightarrow$Elim) $\Gamma, \Delta \vdash f: A \rightarrow B, \quad \Gamma, \Delta \vdash a: A \Rightarrow \Gamma, \Delta \vdash f(a): B$.

Soundness. Hyp.: $\forall \gamma \in [\![\Gamma \vdash \diamond]\!]. \forall \delta \in [\![\Gamma, \Delta \vdash \diamond]\!]\ \gamma.$
$\text{typeErasure}(f)_{\Delta \leftarrow \delta} \in [\![\text{vars}(\Gamma) \vdash A \rightarrow B]\!]\gamma$
$\forall \gamma \in [\![\Gamma \vdash \diamond]\!]. \forall \delta \in [\![\Gamma, \Delta \vdash \diamond]\!]\ \gamma.$
$\text{typeErasure}(a)_{\Delta \leftarrow \delta} \in [\![\text{vars}(\Gamma) \vdash A]\!]\gamma$
Th.: $\forall \gamma \in [\![\Gamma \vdash \diamond]\!]. \forall \delta \in [\![\Gamma, \Delta \vdash \diamond]\!]\ \gamma.$
$\text{typeErasure}(f(a))_{\Delta \leftarrow \delta} \in [\![\text{vars}(\Gamma) \vdash B]\!]\gamma.$

Proof. By Hyp. and by def. ($[\![]\!] \rightarrow$), $\text{typeErasure}(f)_{\Delta \leftarrow \delta}(\text{typeErasure}(a)_{\Delta \leftarrow \delta}) \in [\![\text{vars}(\Gamma) \vdash B]\!]\gamma$.

The thesis follows, since $\text{typeErasure}(f(a))_{\Delta \leftarrow \delta} = \text{typeErasure}(f)_{\Delta \leftarrow \delta}(\text{typeErasure}(a)_{\Delta \leftarrow \delta})$. ▮

$(\forall\text{Intro})\ \Gamma, t \leqslant A, \Delta \vdash b: B, t \notin \text{FTV}(\Delta) \Rightarrow \Gamma, \Delta \vdash \Lambda t \leqslant A.b: \forall t \leqslant A.B.$

Soundness. Hyp.: $\forall \langle \gamma, \iota \rangle \in [\![\Gamma, t \leqslant A \vdash \Diamond]\!].\forall \delta \in [\![\Gamma, t \leqslant A, \Delta \vdash \Diamond]\!]\langle \gamma, \iota \rangle.$
$\text{typeErasure}(b)_{\Delta \leftarrow \delta} \in [\![\text{vars}(\Gamma), t \vdash B]\!]\langle \gamma, \iota \rangle$

Th.: $\forall \gamma \in [\![\Gamma \vdash \Diamond]\!].\forall \delta \in [\![\Gamma, \Delta \vdash \Diamond]\!]\,\gamma.$
$\text{typeErasure}(\Lambda t \leqslant A.b)_{\Delta \leftarrow \delta} \in [\![\text{vars}(\Gamma) \vdash \forall t \leqslant A.B]\!]\,\gamma.$

Applying definitions ($[\![]\!] \neq \varnothing\text{TypeEnv}$) and ($[\![]\!]\forall$) we can rewrite Hyp. and Th., respectively, as:

Hyp.: $\forall \gamma \in [\![\Gamma \vdash \Diamond]\!].\forall \iota \in \text{SAT}.\iota \subseteq [\![\text{vars}(\Gamma), t \vdash A]\!]\langle \gamma, \iota \rangle$
$\Rightarrow \forall \delta \in [\![\Gamma, t \leqslant A, \Delta \vdash \Diamond]\!]\langle \gamma, \iota \rangle.\text{typeErasure}(b)_{\Delta \leftarrow \delta} \in [\![\text{vars}(\Gamma), t \vdash B]\!]\langle \gamma, \iota \rangle$

Th.: $\forall \gamma \in [\![\Gamma \vdash \Diamond]\!].\forall \delta \in [\![\Gamma, \Delta \vdash \Diamond]\!]\gamma.\forall \iota \in \text{SAT}.\iota \subseteq [\![\text{vars}(\Gamma), t \vdash A]\!]\langle \gamma, \iota \rangle$
$\Rightarrow \text{typeErasure}(\Lambda t \leqslant A.b)_{\Delta \leftarrow \delta} = \text{typeErasure}(b)_{\Delta \leftarrow \delta} \in [\![\text{vars}(\Gamma), t \vdash B]\!]\langle \gamma, \iota \rangle.$

Proof. Let $\gamma \in [\![\Gamma \vdash \Diamond]\!], \delta \in [\![\Gamma, \Delta \vdash \Diamond]\!]\gamma, \iota \in \text{SAT}, \iota \subseteq [\![\text{vars}(\Gamma), t \vdash A]\!]\langle \gamma, \iota \rangle$.
By Lemma 3.17 (ValueEnvWeakening), since $t \notin \text{FTV}(\Delta)$, $\delta \in [\![\Gamma, \Delta \vdash \Diamond]\!]\,\gamma \Rightarrow \delta \in [\![\Gamma, t \leqslant A, \Delta \vdash \Diamond]\!]\langle \gamma, \iota \rangle$.
We obtain $\text{typeErasure}(b)_{\Delta \leftarrow \delta} \in [\![\text{vars}(\Gamma), t \vdash B]\!]\langle \gamma, \iota \rangle$ by applying Hyp. to γ, ι, and δ. ▮

$(\forall\text{Elim})\ \Gamma, \Delta \vdash f: \forall t \leqslant A.B, \quad \Gamma \vdash A' \leqslant A[t \leftarrow A'] \Rightarrow \Gamma, \Delta \vdash f\{A'\}: B[t \leftarrow A'].$

Soundness. Hyp.: $\forall \gamma \in [\![\Gamma \vdash \Diamond]\!].\forall \delta \in [\![\Gamma, \Delta \vdash \Diamond]\!]\gamma.$
$\text{typeErasure}(f)_{\Delta \leftarrow \delta} \in [\![\text{vars}(\Gamma) \vdash \forall t \leqslant A.B]\!]\,\gamma$
$\forall \gamma \in [\![\Gamma \vdash \Diamond]\!].[\![\text{vars}(\Gamma) \vdash A']\!]\,\gamma \subseteq [\![\text{vars}(\Gamma) \vdash A[t \leftarrow A']]\!]\,\gamma$

Th.: $\forall \gamma \in [\![\Gamma \vdash \Diamond]\!].\forall \delta \in [\![\Gamma, \Delta \vdash \Diamond]\!]\,\gamma.$
$\text{typeErasure}(f\{A'\})_{\Delta \leftarrow \delta} \in [\![\text{vars}(\Gamma) \vdash B[t \leftarrow A']]\!]\,\gamma.$

We apply definition ($[\![]\!]\forall$) to the first hypothesis, and Lemma 3.18 (TypeSubst) to the second hypothesis and to Th.:

Hyp.: $\forall \gamma \in [\![\Gamma \vdash \Diamond]\!].\forall \delta \in [\![\Gamma, \Delta \vdash \Diamond]\!]\gamma.\forall \iota \in \text{SAT}.$
$\iota \subseteq [\![\text{vars}(\Gamma), t \vdash A]\!]\langle \gamma, \iota \rangle$
$\Rightarrow \text{typeErasure}(f)_{\Delta \leftarrow \delta} = \text{typeErasure}(f\{A'\})_{\Delta \leftarrow \delta} \in [\![\text{vars}(\Gamma), t \vdash B]\!]\langle \gamma, \iota \rangle$ (a)
$\forall \gamma \in [\![\Gamma \vdash \Diamond]\!].[\![\text{vars}(\Gamma) \vdash A']\!]\,\gamma \subseteq [\![\text{vars}(\Gamma), t \vdash A]\!]\langle \gamma, [\![\text{vars}(\Gamma) \vdash A']\!]\,\gamma \rangle$ (b)

Th.: $\forall \gamma \in [\![\Gamma \vdash \Diamond]\!].\forall \delta \in [\![\Gamma, \Delta \vdash \Diamond]\!]\,\gamma.$
$\text{typeErasure}(f\{A'\})_{\Delta \leftarrow \delta} \in [\![\text{vars}(\Gamma), t \vdash B]\!]\langle \gamma, [\![\text{vars}(\Gamma) \vdash A']\!]\,\gamma \rangle$

By (b), $[\![\text{vars}(\Gamma) \vdash A']\!]\,\gamma$ can be ι in (a), yielding: $\text{typeErasure}(f\{A'\})_{\Delta \leftarrow \delta} \in [\![\text{vars}(\Gamma), t \vdash B]\!]\langle \gamma, [\![\text{vars}(\Gamma) \vdash A']\!]\,\gamma \rangle$.

$(\text{Subsump})\quad \Gamma, \Delta \vdash a: A, \quad \Gamma \vdash A \leqslant B \Rightarrow \Gamma, \Delta \vdash a: B.$

Hyp.: $\forall \gamma \in [\![\Gamma \vdash \Diamond]\!].\forall \delta \in [\![\Gamma, \Delta \vdash \Diamond]\!]\,\gamma.\text{typeErasure}(a)_{\Delta \leftarrow \delta} \in [\![\text{vars}(\Gamma) \vdash A]\!]\,\gamma$
$\forall \gamma \in [\![\Gamma \vdash \Diamond]\!].[\![\text{vars}(\Gamma) \vdash A]\!]\gamma \subseteq [\![\text{vars}(\Gamma) \vdash B]\!]\,\gamma$

Th.: $\forall \gamma \in [\![\Gamma \vdash \Diamond]\!].\forall \delta \in [\![\Gamma, \Delta \vdash \Diamond]\!]\,\gamma.\text{typeErasure}(a)_{\Delta \leftarrow \delta} \in [\![\text{vars}(\Gamma) \vdash B]\!]\,\gamma.$

4. CONCLUSIONS

We have presented a fully developed proof of $\beta-\eta-\beta 2-\eta 2$ strong normalization for system *F*-bounded. We have checked every detail and are now quite convinced that system *F*-bounded is strongly normalizing. We hope that the way we have set out our proof will allow the doubtful reader to double check it.

ACKNOWLEDGMENTS

We gratefully acknowledge the anonymous referees for their constructive criticism.

Received June 2, 1995; final manuscript received June 24, 1997

REFERENCES

[Abadi Cardelli 95] Abadi, M., and Cardelli, L. (1995), On subtyping and matching, *in* "ECOOP '95," pp. 145–167.

[Amadio Cardelli 93] Amadio, R., and Cardelli, L. (1993), Subtyping recursive types, *Assoc. Comput. Mach. TOPLAS* **15**(4), 575–631.

[Bruce 94] Bruce, K. B. (1994), A paradigmatic object-oriented programming language: Design, static typing and semantics, *J. Funct. Prog.* **4**(2), 127–206.

[Bruce Longo 90] Bruce, K. B., and Longo, G. (1990), A modest model of records, inheritance and bounded quantification, *Inform. and Comput.* **87**(1/2), 196–240.

[Bruce *et al.* 96] Bruce, K. B., Cardelli, L., Castagna, G., The Hopkins Object Group, Leavens, G. T., and Pierce, B. (1996), On binary methods, *Theory and Practice of Object Systems.* [To appear].

[Bruce *et al.* 95] Bruce, K. B., Schuett, A., and van Gent, R. (1995), PolyTOIL: A type-safe polymorphic object-oriented language, *in* "ECOOP '95."

[Canning *et al.* 89] Canning, P., Cook, W., Hill, W., Mitchell, J. C., and Olthoff, W. (1989), *F*-bounded quantification for object-oriented programming, *in* "Functional Programming Languages and Computer Architecture," pp. 273–280.

[Cardelli *et al.* 94] Cardelli, L., Martini, S., Mitchell, J. C., and Scedrov, A. (1994), An extension of system *F* with subtyping, *Inform. Comput.* **109**(1/2), 4–56.

[Cardelli Wegner 85] Cardelli, L., and Wegner, P. (1985), On understanding types, data abstraction and polymorphism, *ACM Comput. Surveys* **17**(4), 471–522.

[Curien Ghelli 92] Curien, P.-L., and Ghelli, G. (1992), Coherence of subsumption in $F_{\leqslant}$, minimum typing and type checking, *Math. Structures Comput. Sci.* **2**(1), 55–91.

[Curien Ghelli 94] Curien, P.-L., and Ghelli, G. (1994), Decidability and confluence of $\beta\eta\text{top}_{\leqslant}$ reduction in $F\leqslant$, *Inform. Comput.* **109**(1/2), 57–114.

[Eifrig *et al.* 94] Eifrig, J., Smith, S., Trifonov, V., and Zwarico, A. (1994), Application of OOP type theory: State, decidability, integration, *in* "OOPSLA '94."

[Fisher and Mitchell 94] Fisher, K., and Mitchell, J. C. (1994), Notes on typed object-oriented programming, *in* "Theoretical Aspects of Computer Software '94" (M. Hagiya and J. C. Mitchell, Eds.), pp. 844–885, Sendai, Japan, LNCS 789.

[Ghelli 90] Ghelli, G., (1990), "Proof Theoretic Studies about a Minimal Type System Integrating Inclusion and Parametric Polymorphism," Ph.D. thesis, TD-6/90, Dipartimento di Informatica dell'Università di Pisa, Italy.

[Girard 72] Girard, J. Y., (1972), "Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérieur," Thèse de Doctorat d'État, Paris.

[Geuvers 93] Geuvers, H., (1993), "Logics and Type Systems," Ph.D. thesis, Catholic University of Nijmegen.

[Goguen 94] Goguen, H., (1994), "A Typed Operational Semantics for Type Theory," Ph.D. thesis, ECS-LFCS-94-304, University of Edinburgh.

[Katiyar 92] Katiyar, D. (1992), Subtyping *F*-bounded Types, "ANSA Workshop of *F*-bounded quantification, Cambridge." [position paper].

[Katiyar *et al.* 94] Katiyar, D., Luckham, D., and Mitchell, J. C. (1994), A type system for prototyping languages, *in* "POPL'94."

[McAllester *et al.* 95] McAllester, D., Kučan, J., and Otth, D. F. (1995), A proof of strong normalization of F_2, F_ω and beyond, *Inform. and Comput.* **121**(2), 193–200.

[Mitchell 86] Mitchell, J. C. (1986), A type-inference approach to reduction properties and semantics of polymorphic expressions, *in* "11th ACM Conf. on Lisp and Functional Programming;" A revised version in "Logical Foundations of Functional Programming" (G. Huet, Ed.), pp. 199–212, Addison–Wesley, Reading, MA, 1990.

[Mitchell 90a] Mitchell, J. C. (1990), Towards a typed foundation for method specialization and inheritance, *in* "POPL'90," pp. 109–124.

[Mitchell 90b] Mitchell, J. C. (1990), Type systems for programming languages, *in* "Handbook of Theoretical Computer Science," North-Holland, Amsterdam.

[Pierce Turner 94] Pierce, B. C., and Turner, D. (1994), Simple type-theoretic foundations for object-oriented programming, *J. Funct. Prog.* **4**(2), 207–247.

[Tait 76] Tait, W. W. (1967), Intentional interpretation of functionals of finite type I, *J. Symbolic Logic* **32**.